Article

Not peer-reviewed version

Full Coverage Path Planning for Torpedo-type AUVs' Marine Survey Confined in Convex Polygon Area

Ji-Hong Li *, Hyungjoo Kang , Min-Gyu Kim , Mun-Jik Lee , Han-Sol Jin

Posted Date: 24 July 2024

doi: 10.20944/preprints202407.1964.v1

Keywords: Marine survey; CPP (coverage path planning); full coverage; AUVs (autonomous underwater vehicles); convex polygon; path smoothing

Article

Full Coverage Path Planning for Torpedo-type AUVs' Marine Survey Confined in Convex Polygon Area

Ji-Hong Li *, Hyungjoo Kang, Min-Gyu Kim, Mun-Jik Lee, Han-Sol Jin

Autonomous Systems R&D Division, Korea Institute of Robotics and Technology Convergence, Jigok-Ro 39, Nam-Gu, Pohang 37666, Republic of Korea; hjkang@kiro.re.kr (H.K.); zxdwa0817@kiro.re.kr (M.-G.K.); mcklee@kiro.re.kr (M.-J.L.); hsjin@kiro.re.kr (H.-S.J.)

* Correspondence: jhli5@kiro.re.kr; Tel.: +82-54-279-0467

Abstract: In this paper, we present a full coverage path planning (CPP) algorithm for the marine surveys conducted in the convex polygon shaped search area. The survey is supposed to carry out by torpedo-type AUVs (autonomous underwater vehicles). Due to their nonholonomic mechanical characteristics, these vehicles have nonzero minimum turning radius. For any given polygon shaped search area, it can always be partitioned into one or more convex polygons. With this in mind, this paper proposes a novel search algorithm called CbSPSA (Calculation based Shortest Path Search Algorithm) for full coverage of any given convex polygon shaped search area. By aligning the search inter-tracks alongside the edge with the minimum height, we can guarantee the minimum number of the vehicle's turns. In addition, the proposed method can guarantee the planned path is strictly located inside the polygon area without overlapped or crossed path lines, and also has the total path length as short as possible. Considering the vehicle's nonzero minimum turning radius, we also propose a sort of smoothing algorithm to smooth the waypoint path searched by CbSPSA so as for the vehicle to exactly follow it. The smoothed path is also guaranteed to be strictly located inside the polygon. Numerical simulation analyses are also carried out to verify the effectiveness of the proposed schemes.

Keywords: marine survey; CPP (coverage path planning); full coverage; AUVs (autonomous underwater vehicles); convex polygon; path smoothing

1. Introduction

As the name suggests, CPP is the task of determining a path that passes over all points of an area or volume of interest while avoiding obstacles [1]–[4]. In the past decades, this kind of path planning problem has attracted significant research interest in the robotics community, due to its coverage of various practical applications, including lawn mowing [5]–[7], NC pocket machine [6], [8]–[12], vacuum cleaning robot [13]–[16], demining robot [17,18], and automated agricultural harvester [19,20]. Apart from the differences in their working purposes, from the perspective of the robot's mechanical moving restriction, these works can be further classified into two groups [5]: lawn mowing [5]–[7],[17]–[20] and milling problems [6]–[16]. It is worth to mention that since the robot cannot move out the cleaning area such as indoor environment, the vacuum cleaning robot should be classified as milling problem. In addition, some other specific applications [21,22] also should be considered as milling case. In the case of lawn mowing, the cutter head is permitted to mow over non-grass regions, while it cannot exit the pocket area in the milling process. Therefore, full coverage, which is one of the most important issue in CPP, can be easily achieved in the case of lawn mowing. On the contrary, it is difficult, if not impossible, to guarantee this kind of full coverage in the milling problem, especially in the case where the pocket has irregular corner area.

In this paper, we consider the marine survey case where torpedo-type AUVs are utilized to search given area using acoustic sonar devices [23]–[26]. For this kind of marine survey, usually there are two restricting conditions should be carefully considered in practice. One is that the vehicle cannot exit the search area (ex., port and harbor survey), and the second is that the vehicle has nonzero turning radius.

 doi:10.20944/preprints202407.1964.v1

At a glance, since the vehicle cannot exit its search area, this is a milling problem. However, due to its operational characteristics, the sonar sensing area can pass over this bound line. Undoubtedly, this is a significant advantage for us to design a full coverage path for marine vehicles. That said, the full coverage issue still remains as a challenge in this case, due to the fact that the vehicle is restricted to move inside the search area with nonzero turning radius.

Due to the fact that the sonar sensing area can cross over the bound line of search area, in [27], the authors proposed a novel scheme called CbSPSA to search for a sort of optimal full coverage path for any given convex polygon. Indeed, CbSPSA is a kind of geometric solution to search a sort of shortest path in the corner area (around each vertex) of any given convex polygon. In the case of any given polygon, due to the fact that this polygon can always be divided into one or more convex polygons [28], the CPP problem for this polygon is easy to be solved by using CbSPSA. However, in [27], the vehicle is modeled as a particle so that can completely follow the straight line waypoint path searched by CbSPSA.

As mentioned before, for most of marine survey vehicles, they have torpedo-type mechanical appearance, and only have three control inputs (surge force, pitch and yaw moments) to handle their 6DOF 3D motion. Due to this kind of mechanical characteristics, the vehicles have nonzero turning radius. This means that in practice it is impossible for these vehicles to exactly track the straight line waypoint path, and therefore it is difficult to guarantee the vehicles' movements to be strictly restricted inside the search area. For this reason, in this paper we extend the result of [27] to the case where the vehicle has this kind of nonzero turning radius. In addition, some of the algorithms in [27] are corrected and also described in more details in this paper in order for the readers more easily understand the algorithm. Here it should be noted that, for the convenience of discussion, in this paper we only consider the convex polygon case. However, due to the fact that any given polygon can always be partitioned into one or several convex polygons [28], it is straightforward to extend the result in this paper to the general polygon case as in [27].

The remainder of this paper is organized as follows. Section 2 presents some of preliminaries and also clarifies the main objective of this paper. Section 3 includes the main algorithms, both of CbSPSA and smoothing algorithm. Some numerical study is carried out in Section 4 to demonstrate the effectiveness of proposed planning algorithms. Finally, a brief conclusion is summarized in Section 5.

Notations: Throughout the paper, $c\mathcal{P} \in \Re^{2\times n}$ denotes a convex polygon with $p_i \in \Re^2$, $i = 1, \cdots, n$, the vertices of which are arranged in clockwise. $CP(c\mathcal{P}) \in \Re^{3\times N}$ presents the full coverage path of $c\mathcal{P}$ with N the number of the path waypoints[1]. $||\vec{v}||$ denotes the Euclidean norm of vector $\vec{v}$ and $\angle p_i p_j$ presents the azimuth angle of $\overrightarrow{p_i p_j}$. $O(x, r)$ denotes a circle with x the center point and r the radius.

2. Problem Statement

Recently, quite a number of unmanned vehicles have been widely utilized in the various marine surveys. And most of these vehicles, from control engineering perspective, are underactuated systems ([29] and references therein). Due to this kind of mechanical characteristics, these vehicle have to have nonzero steady turning radius [30]. For the convenience of discussion, the parameter $r_m > 0$ is used to denote the minimum steady turning radius.

Most of the marine surveys are aiming to search the specific sea floor area using various sonar devices [23]–[26], and the survey mission is carried out by scanning the area using sonar beam width (or swatch). Here the sonar swath is determined by the device specifications, and varies with the vehicle's altitude from the sea floor. For the convenience of discussion, this paper only considers the simple case where the sonar swath L_s is kept as a constant value. This kind of constant sonar swath can be achieved in practice by forcing the underwater vehicles to keep constant altitude during the survey.

[1] This kind of redefinition of full coverage path is another main upgrade to the authors' previous work [25].

 doi:10.20944/preprints202407.1964.v1

In practice, the search area for most of marine surveys is determined by a series of waypoints marked on the map. By simply connecting these points using straight lines, it is easy to get one or multiple polygons. For any given polygon, it can always be partitioned into one or several convex polygons [28], and the coverage path for this polygon can be easily get by simply integrating all the paths in each of convex polygons [27]. Under this consideration, in this paper we only tackle the simple convex polygon case so that can better convey the main idea of the proposed scheme.

Another interesting issue for this marine survey using acoustic sonar is that, to improve the acquired sonar image quality, the vehicle is usually required to take as less turn numbers (or inter-track numbers) as possible during the survey. Here a inter-track means a long corridor with the width of L_s. With this in mind, the optimal concept in this paper is dedicated to that, for any given search area, the searched coverage path has the minimum turn numbers with the corresponding shortest total path length.

Consequently, for any given $c\mathcal{P}$, the objective of this paper is to search a full coverage path through solving the following three sub-problems:

SP1. Full coverage path with the minimum turn numbers.
SP2. The whole path is strictly located inside the search area with none of overlapped or crossed path lines. Moreover, the total length of the path should be as short as possible.
SP3. The vehicle is allowed to have nonzero turning radius.

The $c\mathcal{P}$ considered in this paper is required to satisfy the following condition.

Assumption 1. For any given $c\mathcal{P}$, each of its interior angles is larger than $2asin(r_m/L_s)$ and also each length of its edges is larger than L_s.

In most of the practical operations, we have $L_s \gg r_m$. That said, appropriate attention still needs to be paid while determining the search area through selecting a series of waypoints on the map.

3. Proposed Method

3.1. Overview

The strategy for full CPP in this paper can be summarized as follows,

- To solve *SP1* for any given $c\mathcal{P}$, the idea in this paper is simple that we pile the inter-tracks alongside a specific edge of $c\mathcal{P}$ so that we can cover $c\mathcal{P}$ with the minimum number of these inter-tracks.
- Apply CbSPSA to solve the *SP2*.
- Also, apply a method similar to CbSPSA combined with the circle whose radius is r_m to round the path at each waypoint (solving *SP3*).

3.2. $c\mathcal{P}$ *Partition:* Solving *SP1* and Some of *SP2*

For any given $c\mathcal{P}$, according to its definition, it is always monotone with respect to each of its edge [31]. For each edge $p_ip_{jmod(i-1,n)}$, $i = 1, \cdots, 7$, where $jmod$ is the same function of mod except that $jmod(0, n) = n$, h_{iM} denotes the maximum height among the vertices p_j, $j = i+1, \cdots, jmod(i-2, n)$ to the edge $p_ip_{jmod(i-1,n)}$ [27]. Here we define $h_M = min\{h_{iM}, i = 1, \cdots, n\}$, from which it is easy to verify that $ceil(h_M/Ls)$ is the minimum number of inter-tracks need to full cover the area $c\mathcal{P}$. Consequently, the strategy in this paper for the vehicle's minimum turn number is straightforward that the vehicle is forced to search the area alongside the inter-tracks parallel to the edge corresponding to h_M so as to minimize the vehicle's turn numbers.

For the convenience of discussion, in this paper all convex polygons take the same form of $[p_1, \cdots, p_n]$, where p_1p_n is the edge corresponding to h_M. Indeed, this kind of expression is of help to simplify the search algorithms proposed in this paper. In this case, we apply $tp = [m, 1, n]$ to denote the specific vertices information for given $c\mathcal{P}$ [27], where p_m is the vertex of h_M to the edge p_1p_n.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 July 2024 doi:10.20944/preprints202407.1964.v1

Moreover, for vertex p_i, if $1 \leq i < m$, then the vertex is said to be on the *LEFT*; otherwise, if $m < i \leq n$, then it is on the *RIGHT*.

In this paper, the vehicle's start and end points are all set at the polygon vertices. More specifically, the start point is set as $incP = [i, dir_i]$ and the end point is $outcP = [j, dir_j]$ with $dir_i, dir_j = \pm 1$, $i, j = 1, \cdots, n$, where the exact position of $x = [k, dir_k]$, $k = 1, \cdots, n$ is defined as following

$$x = p_k + 0.5L_s[cos\alpha_k; sin\alpha_k] \tag{1}$$

where $\alpha_k = \angle p_i p_{jmod(i+dir_k,n)}$.

For any given $incP$ and $outcP$, in order to avoid the possible overlapped or crossed path occurrence, the polygon cP is further partitioned into three parts: $inLet$, cP^c, and $outLet$, as seen in Figure 1. The rule of partition is that: 1) both the end of $inLet$ and the start of $outLet$ are connected to the one of vertices $p_{tp(i)}$, $i \in \{1, 2, 3\}$; 2) if the end of $inLet$ is linked to $p_{tp(1)}$, then the start part of $outLet$ is to $p_{tp(2)}$ or $p_{tp(3)}$, and vice versa. For details of this partition will be discussed later.

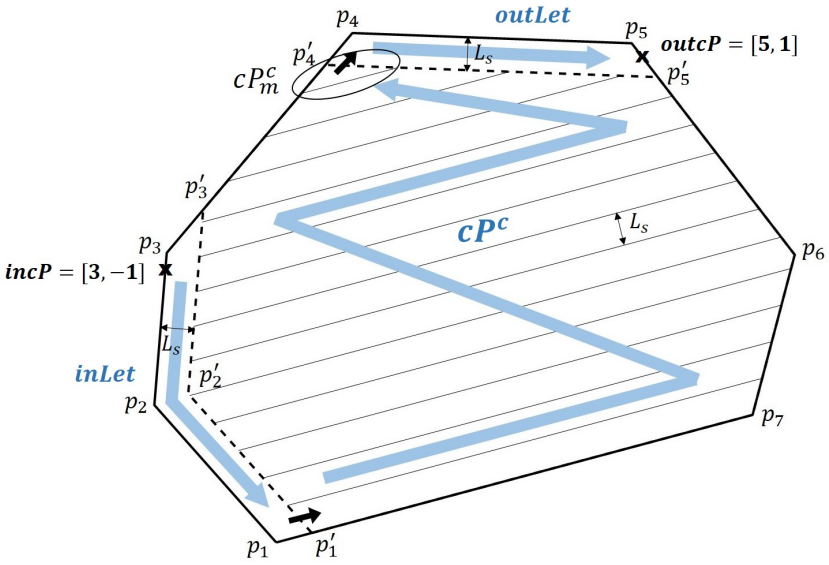


Figure 1. Convex polygon partition with $inLet$, cP^c, and $outLet$ for given $incP = [3, -1]$ and $outcP = [5, 1]$.

According to the partition rule, the overall full coverage for cP can be constructed by

$$CP(cP) = CP(inLet) + CP(cP^c) + CP(outLet). \tag{2}$$

To guarantee the coverage path calculated through (2) to have the minimum turn numbers, we need the following condition to be satisfied.

Assumption 2. Considering cP^c in Figure 1, its tp^c is the same as tp for cP.

According to the partition rule, it is not difficult to verify that even in the case where the *Assumption* 2 is not satisfied, the maximum increasing turn number calculated through (2) will be less than 2 compare to the true minimum turn number. Therefore, in the case $h_M \gg L_s$, which is a very common case in practice, whether *Assumption* 2 is true or not becomes less critical.

3.3. CbSPSA for $CP(cP)$: Solving *SP2*

For any given cP, according to the different locations of $incP$ and $outcP$, there are different partition methods. At first, the different locations of $incP$ and $outcP$ can be divided into two main categories: 1) both of them are on the same side (*LEFT* or *RIGHT*), 2) each on the other side. It is notable that in this paper we only consider the case where $incP$ and $outcP$ are not on the same vertex. In the

 doi:10.20944/preprints202407.1964.v1

case 1), the partition method is straightforward that: if $incP$ is closer to p_m, then $inLet = incP \to p_m$ and $outLet = outcP \to p_1$(on the *LEFT*) or p_n(on the *RIGHT*), and vice versa. In the latter case, also there are two different methods, one is $inLet = incP \to p_m$ and $outLet$ links to $p_1 p_n$ and the other one is vice versa. However, in this case, it's difficult to determine which one is best. In [27], the authors choose the one whose sum of edge lengths is shorter. In this paper, this method is upgraded to that we choose the one whose total length of consequent $CP(cP)$ is shorter.

Consequently, according to (2), simply connecting the $CP(inLet)$, $CP(cP^c)$, and $CP(outLet)$ at each of end and start points, we can search out the full CPP for cP.

Algorithm 1 shows the overall flowchart of CbSPSA for any given convex polygon cP. For given cP, $searchSH(cP)$ return $tp = [m, 1, n]$, details of which are described in the previous subsection 3.2. The function $carvePolygon(cP, tp, incP, outcP)$ carries out the partition of cP as seen in (2) according to given $incP$ and $outcP$). Usually, the result of $carvePolygon(\cdots)$ is not unique, instead it's possible there are several different ways of partitions, among which we will choice the one with the shortest overall waypoint path length. For the remained functions $CP(inLet)$, $CP(outLet)$, and $Search_cPolygon(\cdots)$, they will be described in details in the following subsections.

Algorithm 1: $Search\_Polygon(cP, incP, outcP)$

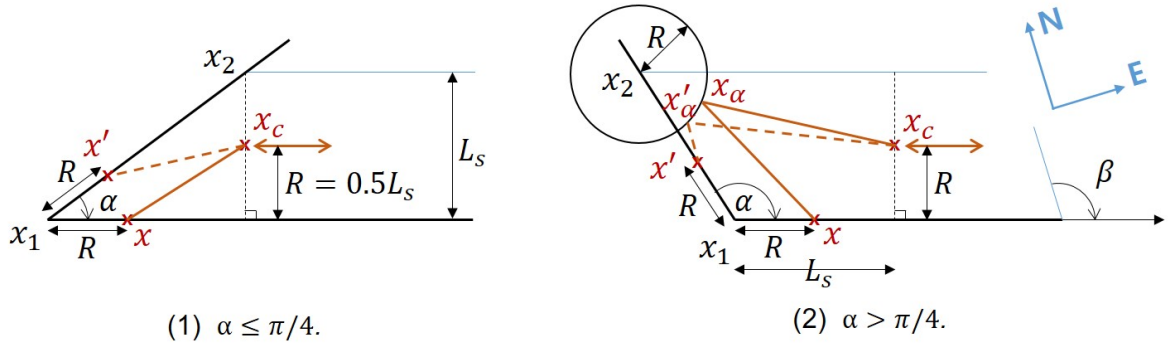
Input: cP, $incP$, $outcP$, **Output:** $CP(cP)$

$tp = searSH(cP)$
$[cP_i^c, inLet_i, OutLet_i]_{i=1}^n = carvePolygon(cP, tp, incP, outcP)$
for $i = 1 : n$
  $CP(cP_i) \leftarrow \emptyset$
  $CP(cP_i) \overset{add}{\Leftarrow} CP(inLet_i)$
  $CP(cP_i) \overset{add}{\Leftarrow} Search_cPolygon(cP_i^c, incP_i^c, outcP_i^c)$
  $CP(cP_i) \overset{add}{\Leftarrow} CP(outLet_i)$
end for
return The shortest path among $CP(cP_i)$, $i = 1, \cdots, n$

3.3.1. CbSPSA for $CP(inLet)$ and $CP(outLet)$

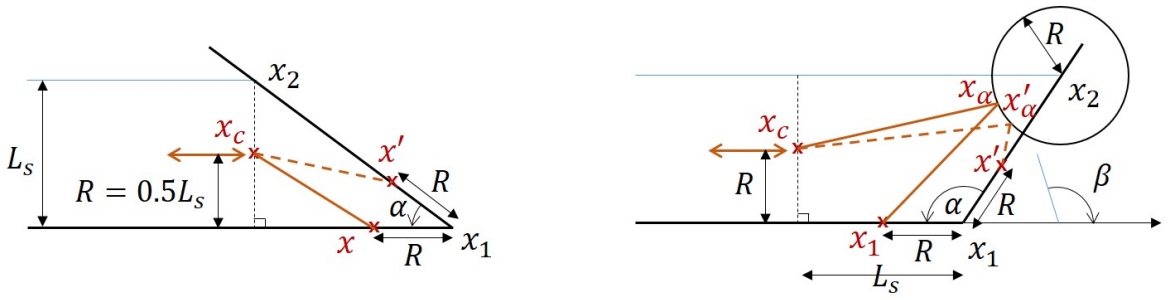
The search algorithms for $inLet$ and $outLet$ are the same except at the end point. At start and end points, the paths are searched using the function $calPoint31(x_1, x_2, \beta, dir)$ as illustrated as in Figure 2, and at the intermediate vertices, using $calPoint32(x, \alpha, \beta, dir)$ which is depicted in Figure 3. It is notable that in the case of $inLet$, at the end point, the waypoints are still searched using $calPoint32(\cdot)$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 July 2024 doi:10.20944/preprints202407.1964.v1



(1) $\alpha \le \pi/4$. (2) $\alpha > \pi/4$.

(a) $dir = -1$ $(Right \rightarrow Left)$.



(1) $\alpha \le \pi/4$. (2) $\alpha > \pi/4$.

(b) $dir = 1$ $(Left \rightarrow Right)$.

Figure 2. Illustration of $calPoint31(x_1, x_2, \beta, dir)$.

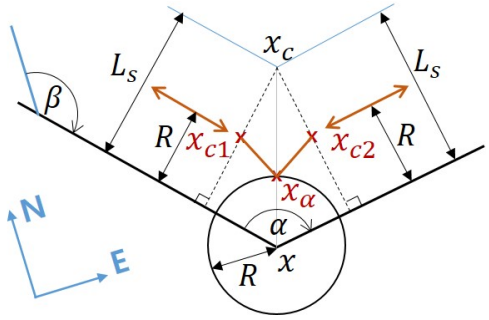


Figure 3. Illustration of $calPoint32(x, \alpha, \beta, dir)$.

3.3.2. CbSPSA for $CP(c\mathcal{P}^c)$

As depicted in Figure 1, $c\mathcal{P}^c$ consists of a number of trapezoids, in other words, inter-tracks. If the vehicle is allowed to move out of the search area as in the case of lawn mowing problem [5]–[7],[17]–[20], then CPP problem becomes straightforward that we simply extend the each centerline to the outside of the search area and just link them one by one. Unfortunately, the vehicle is restricted to move inside the search area and also has nonzero turning radius. In this case, the key issues for CPP are as follows: 1) how to link the centerlines one-by-one in the corner areas, and 2) how to design a shortest pass to cover the area $c\mathcal{P}^c_m$. Accordingly, the proposed search algorithm CbSPSA for $CP(c\mathcal{P}^c)$ is presented as the pseudo-codes in *Algorithm 2* and *Algorithm 3*. For the convenience of discussion, in the algorithm we set $inc\mathcal{P}^c = [m', \pm 1]$ and $outc\mathcal{P}^c = [1, +1]$ or $outc\mathcal{P}^c = [n, -1]$.

 doi:10.20944/preprints202407.1964.v1

In *Algorithm 2*, the state variable $stat$ is defined as in Table 1 and accordingly $c\mathcal{P}_m^c$ is constructed as $c\mathcal{P}_m^c = [p_{L1}, p_2, p'_m, p_4, p_{R1}]$, where $p_2 = p'_{jmod(m-1,n)}$ if $stat[2] = 1$ and $p_4 = p'_{jmod(m+1,n)}$ if $stat[3] = 1$[2]. Also, the parameter dir_i in the algorithm can be determined according to Table 2. Here it is worth to mention that, considering Figure 1 where $inLet$ is connected to p'_1 of $c\mathcal{P}^c$ which is taken as $outc\mathcal{P}^c$ in the algorithm, the direction of $CP(c\mathcal{P}^c)$ should be reversed.

Algorithm 2: $Search\_cPolygon(c\mathcal{P}^c, inc\mathcal{P}^c, outc\mathcal{P}^c)$

Input: $c\mathcal{P}^c$, $inc\mathcal{P}^c$, $outc\mathcal{P}^c$, **Output:** $CP(c\mathcal{P}^c)$
$CP(c\mathcal{P}^c) \leftarrow \emptyset$
$n_c = ceil(h'_M / L_s)$
LQ:=$[p_{L1}, p_{L2}, \cdots, p_{Ln_c}]$, RQ:=$[p_{R1}, p_{R2}, \cdots, p_{Rn_c}]$
for $i = 1 : n_c$
  if $i == 1$
    Set $stat$ with $inc\mathcal{P}^c$, $outc\mathcal{P}^c$, and n_c
    $CP(c\mathcal{P}^c) \overset{add}{\Leftarrow} Search\_cmPolygon(c\mathcal{P}_m^c, stat)$
  else
    X:=last way point of $CP(c\mathcal{P}^c)$
    $CP(c\mathcal{P}^c) \overset{add}{\Leftarrow} calPoint12(X, i, dir_i, LQ, RQ)$
    if $i < n_c$
      $CP(c\mathcal{P}^c) \overset{add}{\Leftarrow} calPoint13(i, dir_i, LQ, RQ)$
    end if
  end if
end for
return $CP(c\mathcal{P}^c)$

As mentioned before, one of main upgrade in this paper is that each waypoint of $CP(c\mathcal{P})$ consists of $wp_i = (x_i, d_i)$ instead of $wp_i = x_i$ as in [27], where $x_i \in \Re^2$ is the horizontal position and the integer parameter d_i indicates an additional information about wp_i. If wp_i is located on the center line of inter-track, then we set $d_i = 2$, otherwise $d_i = -2$. This kind of information will be used in the later smoothing process for $CP(c\mathcal{P})$. On the other hand, in the algorithm, the function $calPoint12(\cdot)$ is similar to $calPoints4(\cdot)$ [27] and $calPoint13(\cdot)$ is to $calPoints5(\cdot)$ in [27], respectively. The difference is that the differentiation criteria is changed from $\alpha \geq \pi/2$ as in Figure 6 in [27] to $\alpha \geq 3\pi/2$. This is for the convenience of path smoothing for the vehicle's nonzero turning radius, details of which will be discussed later.

The *Algorithm 2* in [27] is also upgraded as follows. Here the detailed description of $calPoints21(\cdot)$ and $calPoints22(\cdot)$ are presented in *Appendix A* and *B*. This kind of detailed presentation is supposed to be of help for the readers' in depth understanding of the proposed algorithm. The remainder of $calPoints23(\cdot)$ is similar to $calPoints22(\cdot)$ and $calPoints24(\cdot)$ is to $calPoints21(\cdot)$. The only difference is the search direction.

[2] In this paper we only consider the case where there is maximum of one vertex between $p'_{jmod(m-1,n)}$ and p'_m. Also, maximum of one vertex between p'_m and $p'_{jmod(m+1,n)}$. From the practical point of view, this is also quite a reasonable consideration.

 doi:10.20944/preprints202407.1964.v1

Algorithm 3: $Search\_cmPolygon(c\mathcal{P}_m^c, stat)$

Input: $c\mathcal{P}_m^c, stat$, **Output:** $CP(c\mathcal{P}_m^c)$

$CP(c\mathcal{P}_m^c) \leftarrow \emptyset \overset{add}{\leftarrow} x_1 := inc\mathcal{P}_m^c = [m, stat[0]]$
case 1. $stat[0] = +1, stat[1] = +1$
  $calPoints21(x_1, c\mathcal{P}_m^c, stat)$
case 2. $stat[0] = +1, stat[1] = -1$
  $calPoints22(x_1, c\mathcal{P}_m^c, stat)$
case 3. $stat[0] = -1, stat[1] = +1$
  $calPoints23(x_1, c\mathcal{P}_m^c, stat)$
case 4. $stat[0] = -1, stat[1] = -1$
  $calPoints24(x_1, c\mathcal{P}_m^c, stat)$
end case
return $CP(c\mathcal{P}_m^c)$

Table 1. Definition of the state variable $stat$.

Bytes	Values	Descriptions
$stat[0]$	± 1	Used to determine $inc\mathcal{P}_m^c = [m, stat[0]]$
$stat[1]$	± 1	Indicates the path direction. +1: $p_{L1} \rightarrow p_{R1}$, -1: $p_{R1} \rightarrow p_{L1}$.
$stat[2]$	1,0	Indicates if there is a vertex between p_{L1} and p'_m (clockwise). 1: yes, 0: no
$stat[3]$	1,0	Indicates if there is a vertex between p'_m and p_{R1} (clockwise). 1: yes, 0: no

Table 2. Determination of dir_i in Algorithm 2.

$outc\mathcal{P}^c(1)$	n_c	i	dir_i
1	odd	odd	$R \rightarrow L$
1	odd	even	$L \rightarrow R$
1	even	odd	$L \rightarrow R$
1	even	even	$R \rightarrow L$
n	odd	odd	$L \rightarrow R$
n	odd	even	$R \rightarrow L$
n	even	odd	$R \rightarrow L$
n	even	even	$L \rightarrow R$

3.4. Smoothing $CP(c\mathcal{P})$: Solving SP3

Here we recall some features of $CP(c\mathcal{P})$ searched by CbSPSA. For given $c\mathcal{P}$, since there are total of $n_c - 1$ number of trapezoids, there are same number of center line pieces in $CP(c\mathcal{P})$ [27], each of which corresponds to two adjacent waypoints. Therefore, there are total of $2(n_c - 1)$ number of waypoints in $CP(c\mathcal{P})$ on the center lines, and the remainder ones are not on these lines. This kind of information is encoded to the integer parameter in $wp_i = [x_i, d_i]$. $d_i = +2$ indicates wp_i is on the center line, and $d_i = -2$ means it is not. Furthermore, each pair of adjacent waypoints are connected by straight lines [27]. In order for the vehicles, which have nonzero turning radius, to exactly follow it, this $CP(c\mathcal{P})$ has to go through a specific smoothing process.

The smoothing process proposed in this paper is as the following *Algorithm 4*, and also depicted in Figure 4–6. In the algorithm, the function $checkPoint(x, c\mathcal{P})$ is used to check if the point x is located inside the polygon $c\mathcal{P}$). If it is, then returns TRUE; otherwise, return FALSE. The integer parameter d_i in $fCP(c\mathcal{P})$ is encoded in different way. $d_i = 1$ indicates the path from wp_i is turning clockwise

 doi:10.20944/preprints202407.1964.v1

alongside the circle with r_m the radius until the next waypoint, $d_i = -1$ means anticlockwise, and $d_i = 0$ is the straightforward motion.

There are a few points need to be mentioned here. First, according to the *Assumption 1*, it is not difficult to verify that two points x_i^- and x_i^+, as in the case of Figure 5, cannot be located outside of $c\mathcal{P}$ at the same time. And more interestingly, it is always possible for us to find out a proper value of $\delta\alpha$ so that both of x_i^- and x_i^+ are all located inside $c\mathcal{P}$. Second, consider the case where one of x_{i-1} and x_{i+1} is located inside the circle[3], see Figure 5. In this case, we can easily turn around the circle until the inside point is relocated outside the circle. For example, if x_{i-1} is located inside the circle, then clockwise turning around the circle can easily relocate the point outside of it. Finally, let us consider an exceptional case as seen in Figure 3, which is the combination of Figure 4 and Figure 5. In this case, the path smoothing process can be depicted as in Figure 6, from which we can see that it seems possible that two circles $O(o_{i-1}, r_m)$ and $O(o_i, r_m)$ can cross each other. In this case, it's impossible to search out a path using *Algorithm 4*. With this in mind, in this paper we propose the following proposition.

Algorithm 4: $SmoothingPath(CP(c\mathcal{P}))$

Input: $CP(c\mathcal{P})$, **Output:** $fCP(c\mathcal{P})$

$fCP(c\mathcal{P}) \leftarrow \emptyset \overset{add}{\Leftarrow} (x_1, 0)$
for i=2:(N-1)
  if $d_i == 2$
    Calculate x_i^- or x_{i+1}^+ as in Figure 5
    $fCP(c\mathcal{P}) \overset{add}{\Leftarrow} (x_i^-, -1), (x_i, 0)$ or $(x_{i+1}, +1), (x_{i+1}^+, 0)$
  else if $d_i == -2$
    Calculate x_i^- and x_i^+ according to Figure 6(a)
    while(!$checkPoint(x_i^-, c\mathcal{P})$)
      Increasing $\delta\alpha$ and calculate x_i^- and x_i^+ according to Figure 6(b)
    while(!$checkPoint(x_i^+, c\mathcal{P})$)
      Increasing $\delta\alpha$ and calculate x_i^- and x_i^+ according to Figure 6(c)
    $fCP(c\mathcal{P}) \overset{add}{\Leftarrow} (x_i^-, -1), (x_i^+, 0)$
  end if
end for
$fCP(c\mathcal{P}) \leftarrow \emptyset \overset{add}{\Leftarrow} (x_N, 0)$
return $fCP(c\mathcal{P})$

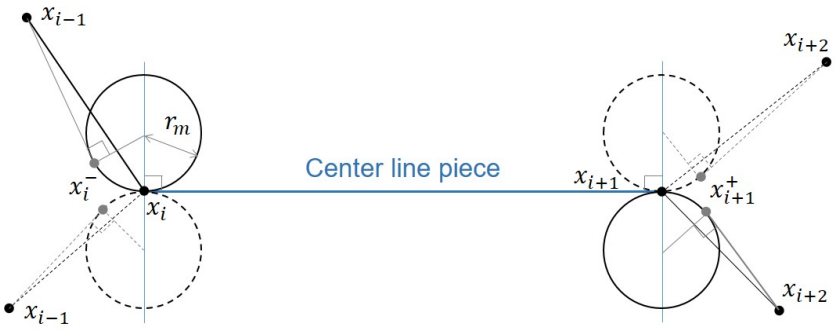


Figure 4. Path smoothing for the waypoint $(x_i, 2)$.

[3] Fortunately, so far the authors have not encounter the case where these two points are inside the circle at the same time. Indeed, with this in mind, the point x_c in Figure 2(a2) and Figure 2(b2) is set away L_s from x_1.

 doi:10.20944/preprints202407.1964.v1

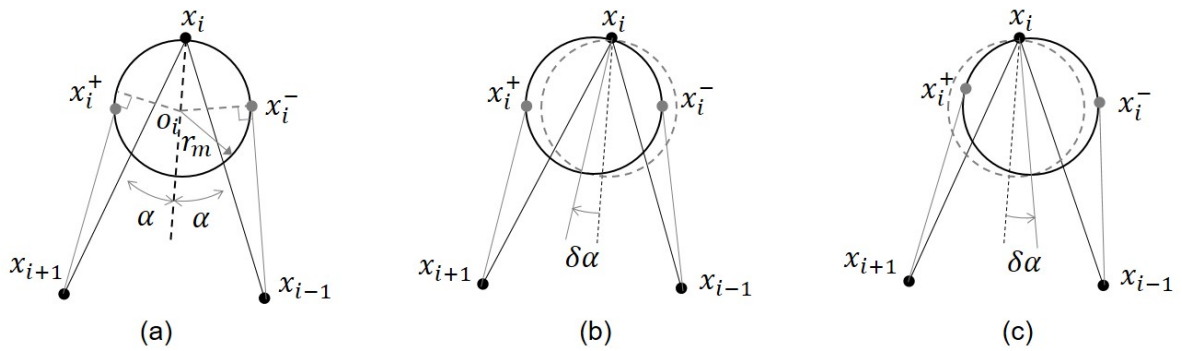


Figure 5. Path smoothing for the waypoint $(x_i, -2)$.

Proposition 1. Consider Figure 6. If r_m satisfies $r_m \leq (\sqrt{5}-2)R$, then we have $||o_i o_{i-1}|| > 2r_m$, $\forall \alpha \in [\alpha_{min}, \pi)$.

Proof of Proposition 1. See Appendix. □

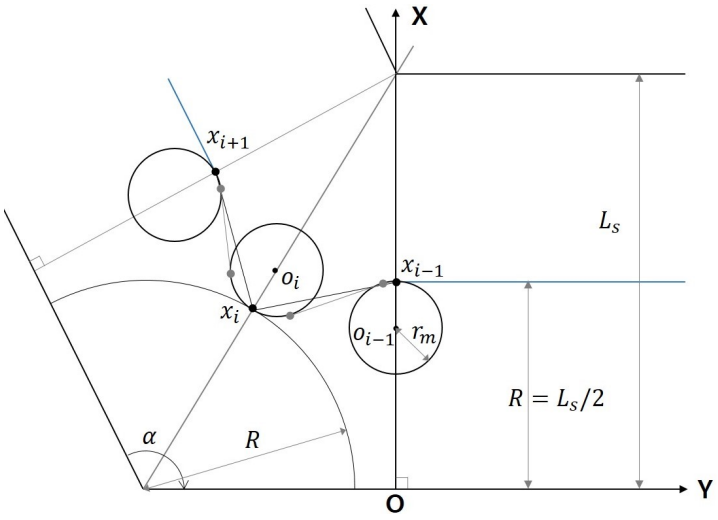


Figure 6. Path smoothing in the case of Figure 3.

Here it's notable that $r_m \leq (\sqrt{5}-2)R$ is a kind of sufficient condition, not a necessary one. With this condition, we can guarantee that all the waypoints in $CP(c\mathcal{P})$ can be properly smoothed so that the vehicle can exactly follow the path and further can guarantee the coverage path planning objective mentioned in Section II to be achieved.

4. Numerical Study

In this section, we carry out some of numerical simulations in MATLAB to further illustrate the proposed path planning algorithm. In the simulation, we consider the convex polygon as $c\mathcal{P} = [p_1; p_2; p_3; p_4; p_5; p_6; p_7] = [181.2m, 676.1m; 850.4m, 82.7m; 1591.2m, 143m; 2661.6m, 1046.8m; 2611.7m, 2406m; 1582m, 3199.6m; 802.3m, 2994.4m]^T$ with $incP = [3, -1]$ and $outcP = [5, 1]$ as seen in Figure 1. For the parameters, the sonar swath is taken as $L_s = 160m$, and the vehicle's minimum turn radius is set as $r_m = 15m$ so that can satisfy the condition in *Proposition 1*.

First, $tp = searchSH(c\mathcal{P})$ returns $tp = [4, 1, 7]$, which means the height from the vertex p_4 to the edge p_1p_7 is the shortest. So the inter-track will be piled alongside p_1p_7. Then, we investigate the convex polygon partition method $carvePolygon(c\mathcal{P}, tp, incP, outcP)$. As mentioned before, since $incP = [3, -1]$ is located on *LEFT* side and $outcP = [5, 1]$ is on the *RIGHT* side, there are total of two partition methods, which means $i = 1, 2$ in *Algorithm 1*. Corresponding results are shown in Figure 7.

 doi:10.20944/preprints202407.1964.v1

In the left case, the total length of $CP(c\mathcal{P})$ is 39382m, and for right one, the length is 38326m. Therefore, *Algorithm 1* return the coverage path as shown in right side in Figure 7.

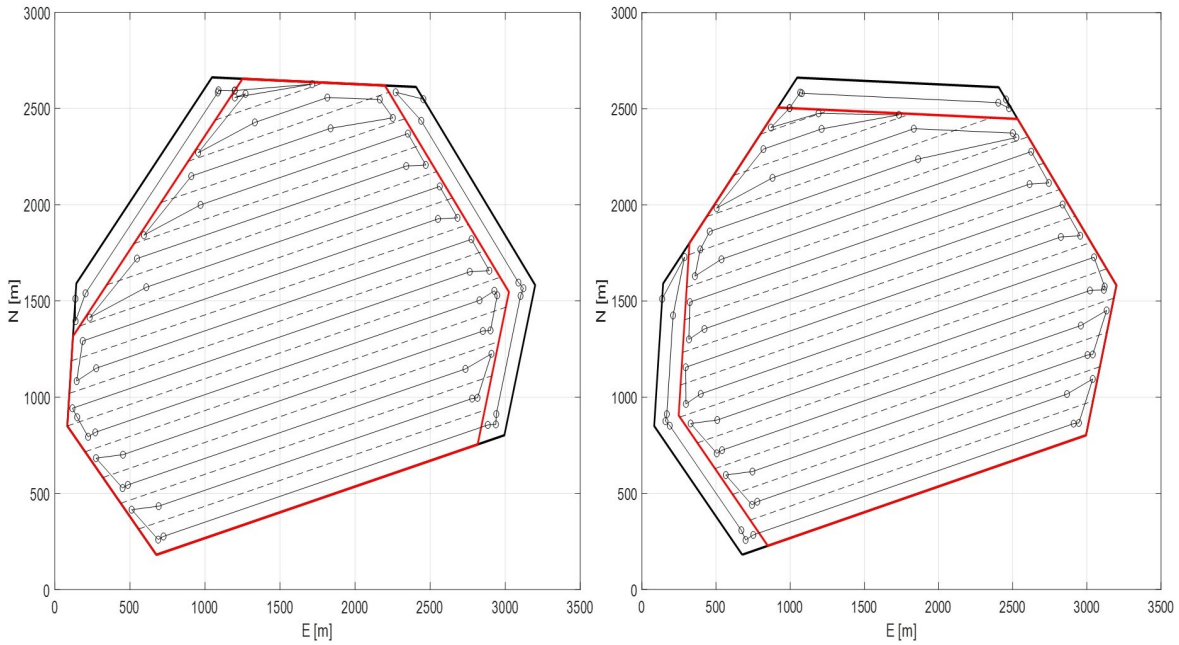


Figure 7. Comparison of different partition methods for given $c\mathcal{P}$.

Now consider the path smoothing algorithm proposed in this paper. For the coverage path $CP(c\mathcal{P})$ searched by *Algorithm 1*, we apply the smoothing method (*Algorithm 4* in subsection 3.4) and the corresponding result is shown in Figure 8. And Figure 9 shows some of the enlarged corner screen of Figure 8. From these results, we can see that the proposed smoothing method can provide a sort of satisfactory smoothing performance while can guarantee the result path all restricted inside the search area. Here it is worth to mention that in the case A as seen in Figure 9(b), since d_1d_2, which is the common tangent between $c\mathcal{P}^c$ and $outLet$, is located inside the polygon $c\mathcal{P}$, the smoothed path has intersection with d_1d_2. In contrast, in the case of B as seen in Figure 9(c), d_3d_4 is the edge of $c\mathcal{P}$, therefore the smoothed path has this kind of form so that to locate the path inside the polygon $c\mathcal{P}$.

 doi:10.20944/preprints202407.1964.v1

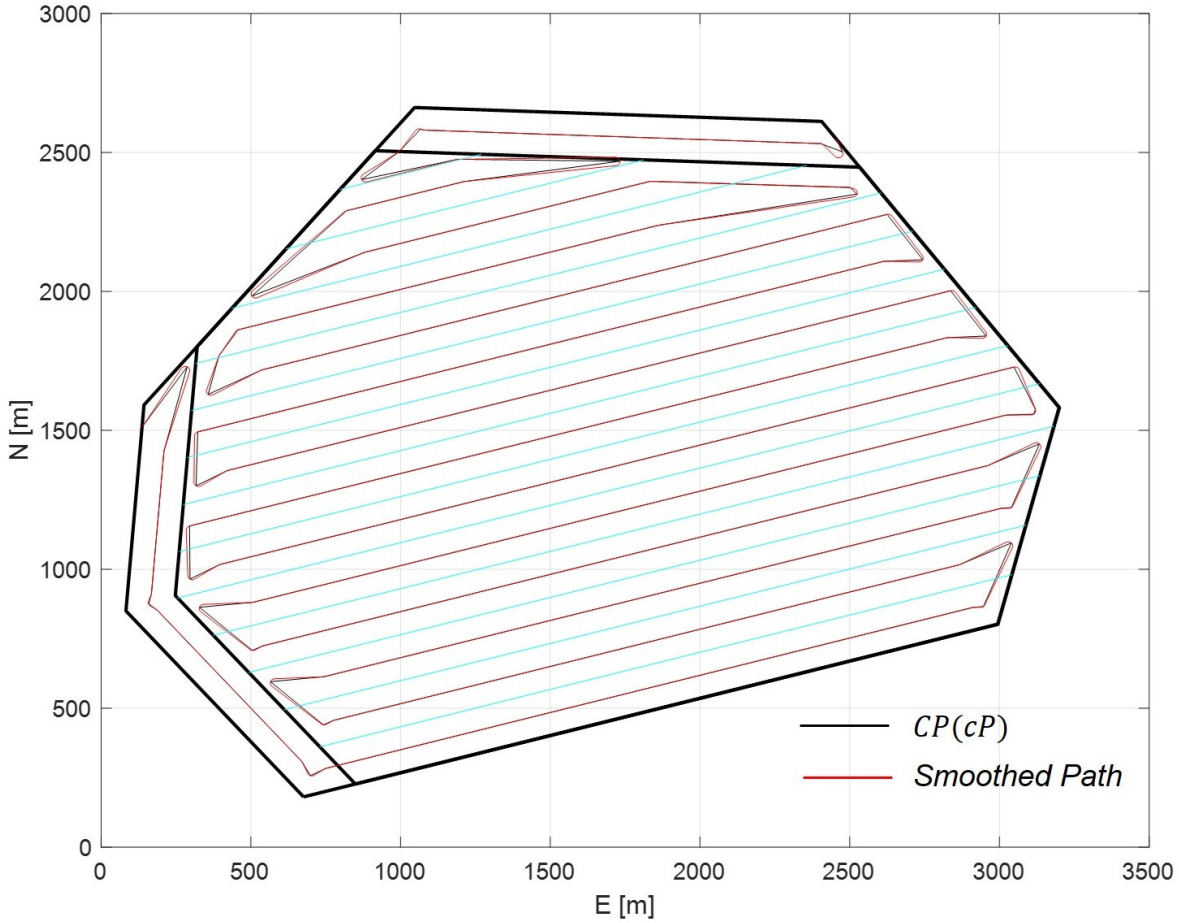


Figure 8. Straight line wayponts path and its smoothing.

 doi:10.20944/preprints202407.1964.v1

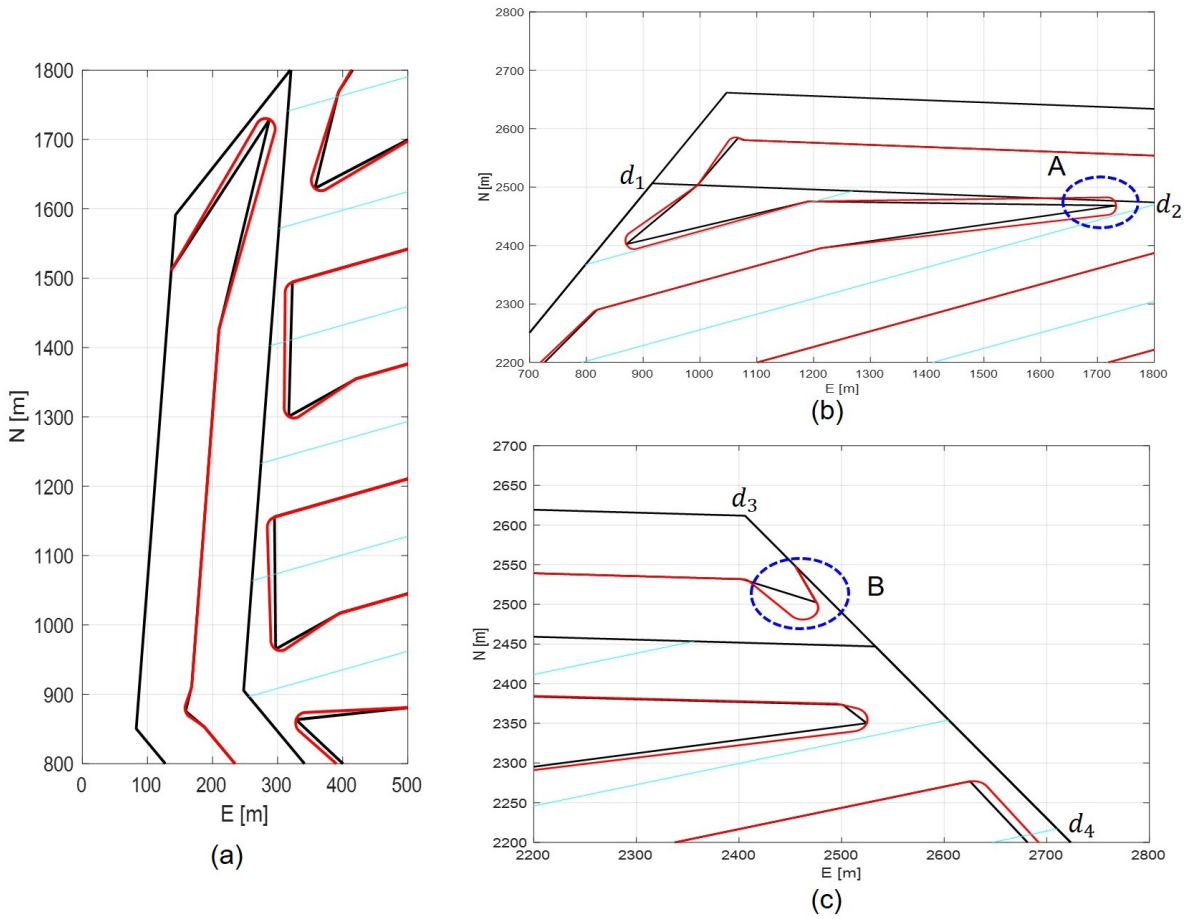


Figure 9. Comparison of straight line path and its smoothing in some corner areas.

Finally, Figure 10 shows the final sonar full coverage result after the vehicle finishes the trajectory following for given smoothed coverage path. Each green line indicates the sonar swath, and from Figure 10, we can see that the sonar swath fully covers the given polygon area $c\mathcal{P}$, while the vehicle keeps moving inside the polygon.

 doi:10.20944/preprints202407.1964.v1

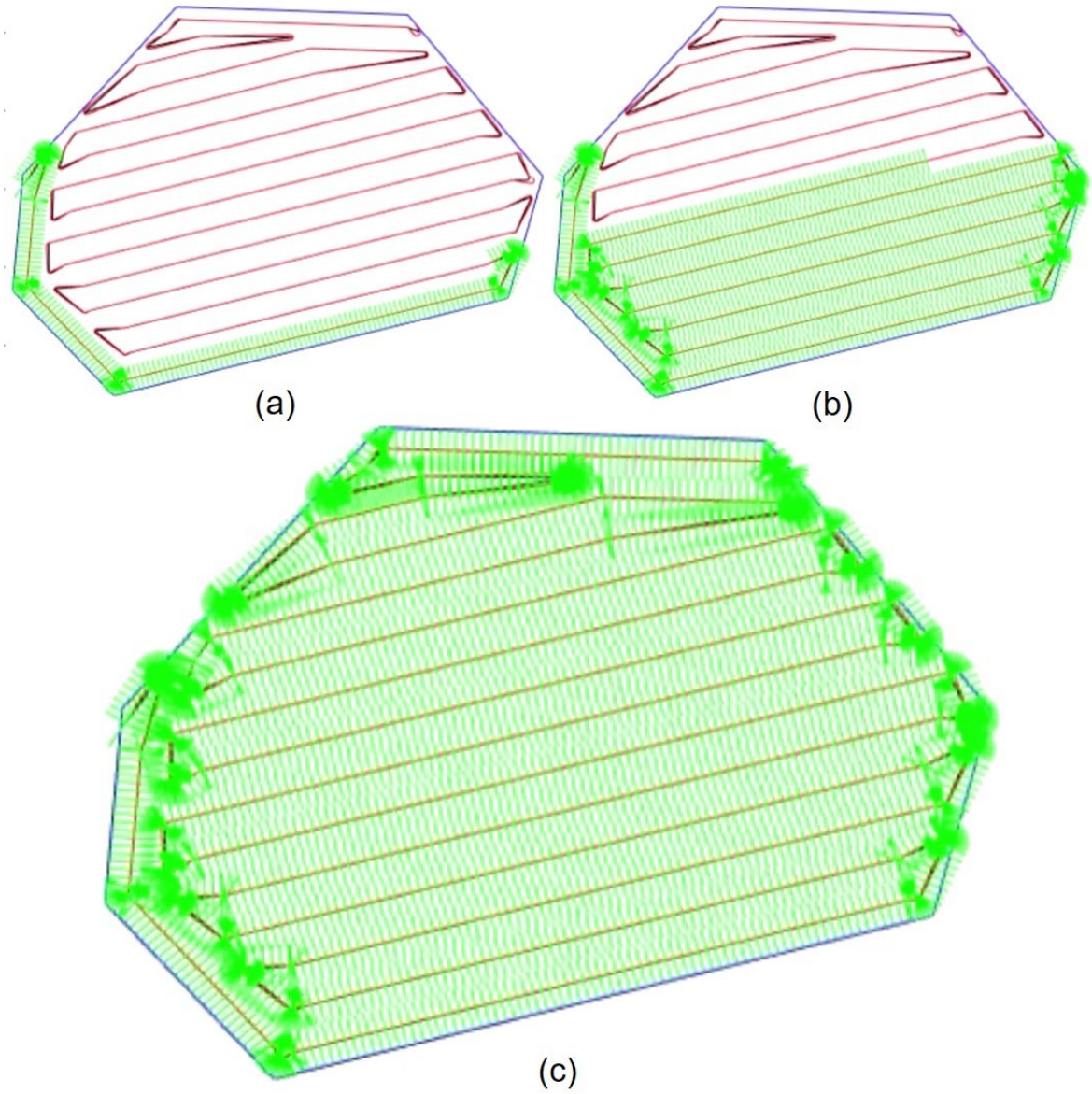


Figure 10. Full coverage of $c\mathcal{P}$ using sonar swath.

5. Conclusion

A full CPP method for marine survey where the vehicles have nonzero minimum turn radius has been presented in this paper. For any given convex polygon search area, it can be partitioned into three parts: $inLet$, $c\mathcal{P}^c$, and $outLet$. At each area, the coverage path is searched using proposed CbSPSA algorithm, and finally integrating these three path can easily construct the final waypoint path. For this straight line waypoint path, we have further proposed a smoothing algorithm so that for the vehicles with nonzero turning radius can exactly track the searched coverage path. Numerical analysis also has been carried out to illustrate the effectiveness of proposed schemes.

The current results in both of this paper and [27] are limited to the convex polygon and some of relatively simple polygon cases. How to extend these results to the case of more general form of polygon search areas might be one of our interesting future works.

Author Contributions: Conceptualization, J.-H.L. and H.K.; methodology, J.-H.L.; software, J.-H.L., H.K., and H.-S.J.; validation, J.-H.L., H.K., M.-G.K., M.-J.L., and H.-S.J; formal analysis, J.-H.L. and H.K.; investigation, J.-H.L.; resources, J.-H.L.; data curation, J.-H.L.; writing—original draft preparation, J.-H.L.; writing—review and editing, J.-H.L.; visualization, J.-H.L. and M.-G.K.; supervision, J.-H.L.; project administration, J.-H.L.; funding acquisition, J.-H.L. All authors have read and agreed to the published version of the manuscript.

 doi:10.20944/preprints202407.1964.v1

Funding: This work was supported in part by the project titled "AUV Fleet and its Operation System Development for Quick Response of Search on Maritime Disasters" of Korea Institute of Marine Science & Technology Promotion (KIMST) funded by the Korea Coast Guard Agency (KIMST-20210547), and in part by Korea Institute of Marine Science & Technology Promotion(KIMST) funded by the Ministry of Oceans and Fisheries(RS-2023-00256122), both in Republic of Korea.

Institutional Review Board Statement: The study did not require ethical approval.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CPP	Coverage path planning
AUV	Autonomous underwater vehicle
CbSPSA	Calculation based shortest path search algorithm
DOF	Degree of freedom

Appendix A. Function $calPoints21(x_1, c\mathcal{P}_m^c, stat)$

$[x_2, b_2] = calPoint1(x_1, p_{R1}, p_{L1})$
if $b_2 > 0$
  if $stat[2] > 0$
    $[x_3, b_3] = calPoint1(x_1, x_2, p_{jmod(m-1,n)})$
    if $b_3 > 0$
      $CP(c\mathcal{P}_m^c) \overset{add}{\leftarrow} (x_3, -2)$
    end if
  end if
  $CP(c\mathcal{P}_m^c) \overset{add}{\leftarrow} (x_2, -2)$
  if $stat[3] > 0$
    $[x_4, b_4] = calPoint1(x_2, p_{R1}, p_{jmod(m+1,n)})$
    if $b_4 > 0$
      $CP(c\mathcal{P}_m^c) \overset{add}{\leftarrow} (x_4, -2)$
    end if
  end if
else if $stat[3] > 0$
  $[x_3, b_3] = calpoint1(x_1, p_{R1}, p_{jmod(m+1,n)})$
  if $b_3 > 0$
    $CP(c\mathcal{P}_m^c) \overset{add}{\leftarrow} (x_3, -2)$
  end if
end if
$x_5 \leftarrow$ last waypoint of $CP(c\mathcal{P}_m^c)$
$x_6 = p_{R1} + 0.5L_s[cos\angle p_{R1}x_5; sin\angle p_{R1}x_5]$
$CP(c\mathcal{P}_m^c) \overset{add}{\leftarrow} (x_6, -2)$
return $CP(c\mathcal{P}_m^c)$

Appendix B. Function $calPoints22(x_1, c\mathcal{P}_m^c, stat)$

$[x_2, b_2] = calPoint1(x_1, p_{L1}, p_{R1})$
if $b_2 > 0$
  if $stat[3] > 0$

 doi:10.20944/preprints202407.1964.v1

$[x_3, b_3] = calPoint1(x_1, x_2, p_{jmod(m-1,n)})$
if $b_3 > 0$
$CP(c\mathcal{P}_m^c) \overset{add}{\Leftarrow} (x_3, -2)$
end if
end if
$CP(c\mathcal{P}_m^c) \overset{add}{\Leftarrow} (x_2, -2)$
$[x_4, b_4] = calPoint1(x_2, p_{L1}, p'_m)$
if $b_4 > 0$
$CP(c\mathcal{P}_m^c) \overset{add}{\Leftarrow} (x_4, -2)$
if $stat[2] > 0$
$[x_5, b_5] = calPoint1(x_4, p_{L1}, p_{jmod(m-1,n)})$
if $b_5 > 0$
$CP(c\mathcal{P}_m^c) \overset{add}{\Leftarrow} (x_5, -2)$
end if
end if
else if $stat[2] > 0$
$[x_5, b_5] = calPoint1(x_2, p_{L1}, p_{jmod(m-1,n)})$
if $b_5 > 0$
$CP(c\mathcal{P}_m^c) \overset{add}{\Leftarrow} (x_5, -2)$
end if
end if
end if
return $CP(c\mathcal{P}_m^c)$

Appendix C. Proof of Proposition 1

From Figure 7, it is easy to get $o_{i-1} = (R - r_m, 0)$ and $o_i = ((R + r_m)sin(\alpha/2), 2R/tan(\alpha/2) - (R + r_m)cos(\alpha/2))$, from which further we have

$$||o_i o_{i-1}||^2 = 2(r_m^2 - R^2) + 2(R + r_m)^2 sin(\alpha/2) - \frac{4R(R+1)}{sin(\alpha/2)} + \frac{4R^2}{sin^2(\alpha/2)}. \tag{A1}$$

With $y = sin(\alpha/2)$, (A1) can be rewritten as following

$$f(y) = 2(r_m^2 - R^2) + 2(R + r_m)^2 y - \frac{4R(R+1)}{y} + \frac{4R^2}{y^2}, \tag{A2}$$

from which it is easy to get

$$f'(y) = 2(R + r_m)^2 + \frac{4R(R+1)}{y^2} - \frac{8R^2}{y^3}, \tag{A3}$$

$$f''(y) = \frac{24R^2}{y^4} - \frac{8R(R + r_m)}{y^3}. \tag{A4}$$

Since $r_m/R \leq \alpha/2 \leq \pi/2$, we have $y \in [r_m/R, 1)$, from which it isn't difficult to verify that, if $r_m < 2R$, then we have $f''(y) < 0$, $\forall y \in [r_m, 1)$. This means that $f'(y)$ is a strictly monotone increasing function. On the other hand, if we set $y = 1$ in (A3), then we have $f'(1) = 2r_m^2 + 8Rr_m - 2R^2$, from which it is easy to verify that if $r_m < (\sqrt{5} - 2)R$, then $f'(1) < 0$. As a result, we can get $f'(y) < f'(1) < 0$, which means that $f(y)$ is monotone decreasing function with $f(1) = 4r_m^2$. Consequently, we can get that if $r_m < (\sqrt{5} - 2)R$, then $||o_i o_{i-1}|| > 2r_m$. This concludes the proof. □

 doi:10.20944/preprints202407.1964.v1

References

1. Choset, H. Coverage for robotics – A survey of recent results. *Annals of Mathematics and Artificial Intelligence* **2001**, *31*, 113–126.
2. Galceran, E.; Carreras, M. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems* **2013**, *61*, 1258–1276.
3. Bormann, R.; Jordan, F.; Hampp, J.; Hagele, M. Indoor Coverage Path Planning: Survey, Implementation, Analysis. In Proceedings of IEEE International Conference on Robotics and Automation, Brisbane, Australia, 2018, 1717–1725.
4. Taua, M. C.; Lisane, B. B.; Ferreira, R. R. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones* **2019**, *3*, doi:10.3390/drones3010004.
5. Arkin, E. M.; Fekete, S. P.; Mitchell, S. B. Approximation algorithms for lawn mowing and milling. *Computational Geometry* **2000**, *17*, 25–50.
6. Huang, Y. Y.; Cao, Z. L.; Oh, S. J.; Kattan, E. U.; Hall, E. L. Automatic Operation for a Robot Lawn Mower. In Proceedings of SPIE 0727, Mobile Robots I, Feb., 2000.
7. Hofmann, M.; Clemens, J.; Stronzek-Pfeifer, D.; Simonelli, R.; Serov, A.; Schettino, S.; Runge, M.; Schill, K.; Buskens, C. Coverage Path Planning and Precise Localization for Autonomous Lawn Mowers. In Proceedings of 6th IEEE International Conference on Robotic Computing, 3–5 Dec., 2022, 238–242.
8. Vosniakos, G. and Papapanagiotou, P. Multiple tool path planning for NC machining of convex pockets without islands. *Robotics and Computer Integrated Manufacturing* **2000**, *16*, 425–435.
9. Chen, X.; Tucker, T. T.; Kurfess, T. R.; Vuduc, R.; Hu, L. Max orientation coverage: efficient path planning to avoid collisions in the CNC milling of 3D objects. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2020, Las Vegas, NV, 6862–6869.
10. Held, M. *On the Computational Geometry of Pocket Maching*, Lecture Notes in Computer Science, *500*, Springer, New York, 1991.
11. Dhanik, S. and Xirouchakis, R. Contour parallel milling tool path generation for arbitrary pocket shape using a fast marching method. *International Journal of Advanced Manufacturing Technology* **2010**, *50*, 1101-1111.
12. Venkatesh, R.; Vijayan, V.; Parthiban, A.; Sathish, T.; Chandran, S. S. Comparison of different tool path pocket milling. *International Journal of Mechanical Engineering and Technology* **2018**, *9*, 922–927.
13. Yasutomi, F.; Yamada, M.; Tsukamoto, K. Cleaning robot control. In Proceedings of IEEE International Conference on Robotics and Automation, 1988, Philadelphia, PA, 1839–1841.
14. Hofner, C. and Schmidt, G. Path planning and guidance techniques for an autonomous mobile cleaning robot. *Robotics and Autonomous* **1995**, *14*, 199–212.
15. Prassler, E.; Ritter, A.; Schaeffer, C.; Fiorini, P. A Short History of Cleaning Robots. *Autonomous Robots* **2000**, *9*, 211–226.
16. Miao, X.; Lee, J.; Kang, B. Y. Scalable Coverage Path Planning for Cleaning Robots Using Rectangular Map Decomposition on Large Environments. *IEEE Access* **2018**, *6*, 38200-38215.
17. Acar, E. U.; Choset, H.; Zhang, Y.; Schervish, M. Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods. *The International Journal of Robotics Research* **2003**, *22*, 441–466.
18. Seder, M. and Petrovic, I. Complete coverage path planning of mobile robots for humanitarian demining. *Industrial Robot* **2012**, *39*, 484–493.
19. Stoll, A. and Kutzbach, D. Guidance of a Forage Harvester with GPS. *Precision Agriculter* **2000**, *2*, 281–291.
20. Oksanen, T. and Visala, A. Coverage Path Planning Algorithms for Agricultural Field Machines. *Journal of Field Robotics* **2009**, *26*, 651–668.
21. Wong, S. C. and MacDonald, B. A. A topological coverage algorithm for mobile robots. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2003, Las Vegas, NV, 1685–1690.
22. Tang, J.; Sun, C.; Zhang, X. MSTC*:Multi-robot Coverage Path Planning under Physical Constraints. In Proceedings of 2021 IEEE International Conference on Robotics and Automation, Xi'an, China, 2021, 2518–2524.
23. Paull, L.; Saeedi, S.; Seto, M.; Li, H. Sensor-Driven Online Coverage Planning for Autonomous Underwater Vehicles. *IEEE/ASME Transactions on Mechatronics* **2013**, *18*, 1827–1838.

 doi:10.20944/preprints202407.1964.v1

24. Sun, B.; Zhu, D.; Tian, C.; Luo, C. Complete Coverage Autonomous Underwater Vehicles Path Planning Based on Glasius Bio-Inspired Neural Network Algorithm for Discrete and Centralized Programming. *IEEE Transactions on Cognitive and Developmental Systems* **2019**, *11*, 73–84.
25. Yordanova, V. and Gips, B. Coverage Path Planning With Trackn Spacing Adaptation for Autonomous Underwater Vehicles. *IEEE Robotics and Automation Letters* **2020**, *5*, 4774–4780.
26. Bagnitckii, A.; Inzartsev, A.; Pavin, A. Planning and correction of the AUV coverage path in real time. In Proceedings of 2017 IEEE Underwater Technology, Busan, Korea, Feb., 2017, DOI:10.1109/UT.2017.7890299.
27. Li, J. H.; Kang, H.; Kim, M. G.; Jin, H.; Lee, M. J.; Cho, G. R.; Bae, C. Full Coverage of Confined Irregular Polygon Area for Marine Survey. *IEEE Access* **2023**, *11*, 92200–92208.
28. Stefan, H. and Kurt, M. Fast triangulation of simple polygons. *Foundations of Computation Theory* **1983**, Lecture Notes in Computer Science. Berlin, Heidelberg, Springer, 158: 207–218.
29. Li, J. H. 3D trajectory tracking of underactuated non-minimum phase underwater vehicles. *Automatica* **2023**, *155*, 111149.
30. Fossen, T. I. *Guidance and Control of Ocean Vehicles*. John Wiley & Sons Ltd., chichester, UK.
31. Heckbert, P. S. *Testing the Convexity of a Polygon*. Academic Press, London, UK.